



# dummy-Notiz-Ubuntu-v03

Jan Unger

24. Dezember 2019

Das Basis-Programm von LaTeX ist TeX und wurde von Donald E. Knuth an der Stanford University entwickelt. Auf TeX aufbauend entwickelte Leslie Lamport Anfang der 1980er Jahre LaTeX, eine Sammlung von TeX-Makros, die die Benutzung für den durchschnittlichen Anwender gegenüber TeX vereinfachten und erweiterten. Der Name LaTeX ist eine Abkürzung für Lamport TeX.

Quelle: Wikipedia <sup>1</sup>



---

<sup>1</sup><https://de.wikipedia.org/wiki/LaTeX>

# Inhalt

<b>1</b>	<b>dummy-Notiz-Ubuntu-v03</b>	<b>1</b>
1.1	L <sup>A</sup> T <sub>E</sub> X	1
1.1.1	Zitieren	1
1.1.2	Quellcode	1
1.1.3	Text und Absatz	1
1.1.4	Listen	2
1.1.5	Referenz und Links	3
1.1.6	Bilder	4
1.1.7	Tabelle	5
1.1.8	Formel	7
1.1.9	Textauszeichnung	8
1.1.10	Farben	8
1.1.11	Zusammenfassung	9
1.2	Neu	9
1.3	Readme	10
1.3.1	Hinweis	10
1.3.2	Projekt erstellen	10
1.3.3	Software	11
1.3.4	Repository von Github downloaden	11
1.3.5	Neues Repository auf Github anlegen	12
1.3.6	Markdown Dokumente - Notizen verfassen	13
1.3.7	Bilder optimieren	15
1.3.8	Backup	16
1.4	Schnellstart	16
1.4.1	Git konfigurieren	16
1.4.2	Projekt mit GitHub einrichten	17
1.4.3	Projekt von GitHub downloaden	17
1.4.4	Projekt lokal einrichten	17
1.4.5	Git Workflow	18
1.4.6	git log - history	19
1.4.7	Branch neu erstellen - PROJEKT bearbeiten	19
1.4.8	git tag - Version erstellen	20

1.4.9	git blame - Wer hat was und wann geändert?	20
1.4.10	Datei umbenennen o. löschen - .git löschen	20
1.4.11	Versionskonflikt lösen	20
1.4.12	Lokales Wiederherstellen	21
1.4.13	Repository Wiederherstellen	22
1.4.14	Git Backup - Repository ohne einen Workspace	23
1.4.15	lokales Repository clonen	24
1.5	Git Workflow - Bereiche	24
1.6	Wiederherstellen	25
1.6.1	Ordner für Experimente erstellen - löschen	25
1.6.2	bestehendes Repository clonen	25
1.6.3	Arbeitsverzeichnis bearbeiten	25
1.6.4	Wiederherstellen: Repository in ein temp. Verzeichnis klonen	26
1.6.5	Wechsel auf den gewünschten Git-Branch	26
1.6.6	Verschiebe .git in den Workspace der alten Versionsverwaltung	27
1.6.7	Ergebnis prüfen	27
1.7	Repository Versionen vergleichen	27
1.7.1	Ordner für Experimente erstellen - löschen	27
1.7.2	lokales Repository	27
1.7.3	Github Repository	27
1.7.4	Backup Repository	28
1.7.5	Ergebnis prüfen	29
1.7.6	Projekt Inhalt	30
1.7.7	build - Versionen erstellen	30
1.8	Markdown - Spickzettel	31
1.8.1	Quellenangabe	31
1.8.2	Listen	31
1.8.3	Anführungszeichen	31
1.8.4	Bilder - Abbildungen	32
1.8.5	Tabelle	32
1.8.6	Mathe	32
1.8.7	Texthervorhebung	32
1.8.8	Code	32
1.8.9	Links	33

1.8.10 Absätze . . . . .	33
--------------------------	----

# 1dummy-Notiz-Ubuntu-v03

## 1.1 L<sup>A</sup>T<sub>E</sub>X

### 1.1.1 Zitieren

Literaturlistenverwaltungsprogramm: JabRef

Zitat: vgl. [Mon16] u. [Kof17]

### 1.1.2 Quellcode

Hallo Welt (vgl. Quellcode 1.1).

**Quelltext 1.1:** Hallo Welt

```
#include <iostream>
int main(){
    std::cout << "Hallo Welt!" << std::endl;
}
```

hallowelt.cpp (vgl. Quellcode 1.2).

**Quelltext 1.2:** hallowelt.cpp

```
/* hallowelt.cpp */
#include <iostream>
int main(){
    std::cout << "Hallo Welt!" << std::endl;
}
```

L<sup>A</sup>T<sub>E</sub>X-Syntax (vgl. Quellcode 1.3).

**Quelltext 1.3:** L<sup>A</sup>T<sub>E</sub>X-Syntax

```
\subsection{Quellcode}
\label{quellcode}
```

### 1.1.3 Text und Absatz

Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist, und weshalb man die Lust anklagt und den Schmerz lobet, so will ich Euch Alles

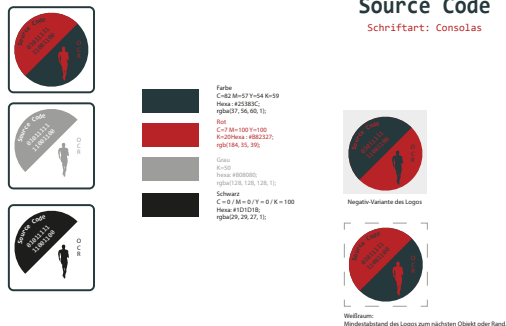
eröffnen und auseinander setzen, was jener Begründer der Wahrheit und gleichsam Baumeister des glücklichen Lebens vornehmen, wenn er nicht einen Vortheil davon erwartete. Wer dürfte aber wohl Den tadeln, der nach einer Lust verlangt, welcher keine Unannehmlichkeit folgt, oder der einem Schmerze ausweicht, aus dem keine Lust hervorgeht?

Dagegen tadelt und hasst man mit Recht Den, welcher sich durch die Lockungen einer gegenwärtigen Lust erweichen und verführen lässt, ohne in seiner blinden Begierde zu sehen, welche Schmerzen und Unannehmlichkeiten seiner deshalb warten. Gleiche Schuld von sich weisen darf. Deshalb trifft der Weise dann eine Auswahl, damit er durch Zurückweisung einer Lust dafür eine grössere erlange oder durch Uebernahme gewisser Schmerzen sich grössere erspare.

Deshalb trifft der Weise dann eine Auswahl, damit er durch Zurückweisung einer Lust dafür eine grössere erlange oder durch Uebernahme gewisser Schmerzen sich grössere erspare.

## 1.1.4 Listen

1. Beschreibung
2. Bild



3. Tabelle

Control Board Label	Wire Color	Signal
VCC	Red	Motor +
GND	Black	GND

#### 4. Werte

- **Temperature1:** 30
- **M1/M2 Amps:** 0.00

#### 1. Aufzählung

#### 2. Aufzählung

#### 3. Clone the code repo:

- a) git clone <https://github.com/ju-bw/notizenDummyWin10-v03.git>
- b) git checkout master
- c) git pull

- Punkt
- Punkt

Direct downloadable Raspbian Buster image:

<https://downloads.raspberrypi.org/raspbian/images/raspbian-2019-07-12/2019-07-10-raspbian-buster.zip>

### 1.1.5 Referenz und Links

- Link <https://www.ju1.eu/>.
- Website.
- File (siehe [/Readme.pdf](#))
- Bild (siehe Abb. [1.1](#))
- Tabelle (siehe Tab. [1.2](#))
- Kapitel (siehe Kap. [1.1](#))
- Textfußnote <sup>1</sup>

---

<sup>1</sup>Fußnote

1.1.6 Bilder

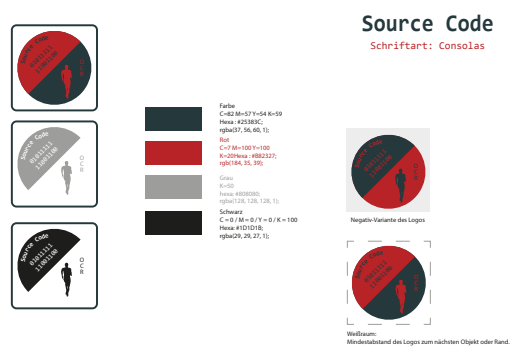
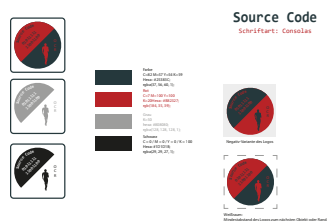


Abb. 1.1: Bild



Logo in Negativ, Grau u. Schwarz (siehe Abb. 1.2)



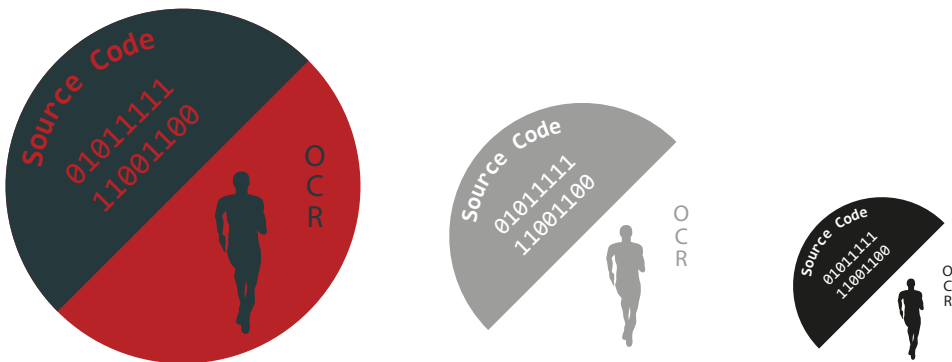


Abb. 1.2: Logo in Negativ, Grau u. Schwarz  
Quelle: <https://www.ju1.eu/>

Logo Details (siehe Abb. 1.3)

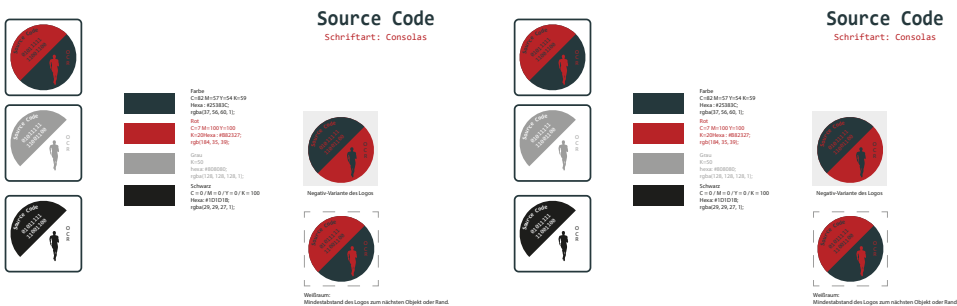


Abb. 1.3: Logo Details

1.1.7 Tabelle

Tabelle (vgl. Tab. 1.1).




Nr.	Vorgehen
1	Aktuellen Forschungsstand recherchieren
2	Methoden entwickeln
3	Schlussfolgerung aufstellen

Tab. 1.1: Tabelle

Teil	Beschreibung
Batterie	Versorgt das System mit Strom. Hat einen unregulierten Spannungsbereich von ca. 11,5 V - 16,75 V je nach Ladezustand
Schalter	mechanische Trennung der elektrischen Energie zum Rest des Roboters

Control Board Label	Wire Color	Signal
VCC	Red	Motor +
GND	Black	GND

Control Board pin	cable wire color
1. GND	Black
2. 5V	Red

Artikel	Ref	Menge	Bild	Artikel	Ref	Menge	Bild
Battery	logo	1		Tamiya Connectors	logo	1	
Battery Charger	logo	1					

**Tab. 1.2:** Komponenten

Pin	Description	Color
1	+5V DC Power	Red
2	GND	Black

**Tab. 1.3:** Pinbelegung

Item	Ref	Board Ref
4.7K 1/4 Watt Res	E7	R1
10K 1/2 Watt Res	E10	R2
100nF Cap	E11	C1-17

Tab. 1.4: Resistor/Capacitor reference

Variable	Physical description
$d_1$	Horizontal distance
$d_2$	Vertical distance

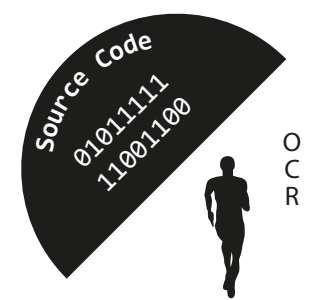


Abb. 1.4: Physical distance

1.1.8 Formel

$d_1 = 7.254mm \quad d_2 = d_3 = 10.5mm \quad d_4 = 10.073mm$  (1.1)

$1 \leq 2 \geq 0 \neq 4, \quad 1 \ll 10^{20} \gg 10^{-5} \pm$  (1.2)

$a \cdot b \quad a \times b \quad \frac{x}{2} \quad a_1 \quad a^2 \quad \begin{pmatrix} a \\ b \end{pmatrix} \quad \sqrt{x} \quad bzw. \quad \sqrt[n]{x}$  (1.3)

$\sum_{i=1}^n i = \frac{n(n+1)}{2}$  (1.4)

$$\prod_{i=1}^{n+1} i = 1 \cdot 2 \cdot \dots \cdot n \cdot (n+1) \quad (1.5)$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0 \quad (1.6)$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad (1.7)$$

### 1.1.9 Textauszeichnung

**fetter Text**

**\*\*Hinweis\*\*** Diese haben NICHT die gleiche Farbe und Pinbelegung wie die Antriebsmotoren!

### 1.1.10 Farben

Apricot Aquamarine Apricot

Apricot Cyan Mahogany ProcessBlue SpringGreen

Aquamarine Dandelion Maroon Purple Tan

BitterSweet DarkOrchid Melon RawSienna TealBlue

Black Emerald MidnightBlue Red Thistle

Blue ForestGreen Mulberry RedOrange Turquoise

BlueGreen Fuchsia NavyBlue RedViolet Violet

BlueViolet Goldenrod OliveGreen Rhodamine VioletRed

Brickred Gray Orange RoyalBlue White

Brown Green OrangeRed RoyalPurple WildStrawberry

BurntOrange GreenYellow Orchid RubineRed Yellow

CadetBlue JungleGreen Peach Salmon YellowGreen

CarnationPink Lavender Periwinkle SeaGreen YellowOrange

Cerulean LimeGreen PineGreen Sepia

CornflowerBlue Magenta Plum SkyBlue

### 1.1.11 Zusammenfassung

Es gibt ein paar wichtige Dinge, die beachtet werden sollten, wie bei jedem Projekt, bei dem mit Batterie oder elektrischem Strom gearbeitet wird:

**DIE BATTERIE.**

## 1.2 Neu

## 1.3 Readme

Erstellt Websites & L<sup>A</sup>T<sub>E</sub>XPDFs mit Markdown und Pandoc.

Sed passt die L<sup>A</sup>T<sub>E</sub>X-Syntax an.

Versionsverwaltung: Git

### 1.3.1 Hinweise

Projekt getestet unter Ubuntu 18.04.2 LTS.

### 1.3.2 Projekt erstellen

Das Script «THEMA-umbenennen.sh» sucht und ersetzt THEMA.

ACHTUNG: Script außerhalb vom Projektordner ausführen.

```
# Github Repository downloaden
REPOSITORY="dummy-notizenUbuntu-v03"
ADRESSE="https://github.com/ju1-eu"
git clone $ADRESSE/$REPOSITORY.git

# Script anpassen
vi THEMA-umbenennen.sh
    # file: THEMA - Suchen und Ersetzen
    ./THEMA-umbenennen.sh

+++ Erste Schritte +++
$ cd neue-Notiz/
# files anpassen
scripte/sed.sh
    - codelanguage:    HTML5, Python, Bash, C, C++, TeX
    - CMS Server Pfad: https://www.ju1.eu/#
    - Bildformat:      pdf, svg, png, jpg

projekt.sh
    - Backupverzeichnis

content/metadata.tex
    - Datum, Titel, Autor

$ ./projekt.sh
    # Projekt erstellen"
```

### 1.3.3 Software

Pandoc: <https://pandoc.org/installing.html>

LaTeX: <https://www.tug.org/texlive/acquire-netinstall.html>

Editor:

<https://code.visualstudio.com/download>

<https://atom.io/>

Git: <https://git-scm.com/downloads>

```
# Shell $
lsb_release -a
# Ubuntu 18.04.3 LTS - Codename: bionic
pandoc -v
# pandoc 1.19.2.4
tex -v
# TeX 3.14159265 (TeX Live 2017/Debian)
git --version
# git version 2.17.1
```

kdiff3: <http://kdiff3.sourceforge.net/>

Imagemagick: <https://www.imagemagick.org/script/download.php#windows>

### 1.3.4 Repository von Github downloaden

<https://github.com/ju1-eu>

**Repository Name** = dummy-notizenUbuntu-v03

```
# Shell $
```

```
# Github Repository downloaden
REPOSITORY="dummy-notizenUbuntu-v03"
ADRESSE="git@github.com:ju1-eu"
git clone $ADRESSE/$REPOSITORY.git
```

```
# Backup Repository clonen
repos_HD="/media/jan/virtuell/repos/notizenUbuntu"
REPOSITORY="dummy-notizenUbuntu-v03"
git clone $repos_HD/$REPOSITORY.git
```

```
repos_USB="/media/jan/usb/repos/notizenUbuntu"
REPOSITORY="dummy-notizenUbuntu-v03"
git clone $repos_USB/$REPOSITORY.git
```

```
repos_RPI4="smb://rpi4.local/nas/repos/notizenUbuntu"  
REPOSITORY="dummy-notizenUbuntu-v03"  
git clone $repos_RPI4/$REPOSITORY.git
```

### 1.3.5 Neues Repository auf Github anlegen

<https://github.com/ju1-eu>

**Repository Name** = dummy-notizenUbuntu-v03

```
# lokales Repository: HEAD -> master  
git init # rm -rf .git  
git add .  
git commit -m"Projekt init"  
  
# Github Repository: origin/master  
# anpassen  
REPOSITORY="dummy-notizenUbuntu-v03"  
ADRESSE="git@github.com:ju1-eu"  
git remote add origin $ADRESSE/$REPOSITORY.git  
git push --set-upstream origin master  
  
# backup Repository: backupUSB/master  
# anpassen  
repos_USB="/media/jan/usb/repos/notizenUbuntu"  
REPOSITORY="dummy-notizenUbuntu-v03"  
LESEZEICHEN="backupUSB"  
git clone --no-hardlinks --bare . $repos_USB/$REPOSITORY.git  
git remote add $LESEZEICHEN $repos_USB/$REPOSITORY.git  
git push --all $LESEZEICHEN  
  
# backup Repository: backupRPI4/master  
# anpassen  
repos_RPI4="smb://rpi4/nas/repos/notizenUbuntu"  
REPOSITORY="dummy-notizenUbuntu-v03"  
LESEZEICHEN="backupRPI4"  
git clone --no-hardlinks --bare . $repos_RPI4/$REPOSITORY.git  
git remote add $LESEZEICHEN $repos_RPI4/$REPOSITORY.git  
git push --all $LESEZEICHEN  
  
# Backup Repository: backupHD/master  
repos_HD="/media/jan/virtuell/repos/notizenUbuntu"  
REPOSITORY="dummy-notizenUbuntu-v03"  
LESEZEICHEN="backupHD"  
git clone --no-hardlinks --bare . $repos_HD/$REPOSITORY.git
```



```
git remote add $LESEZEICHEN $repos_HD/$REPOSITORY.git
git push --all $LESEZEICHEN
```

## Git Befehle

```
# Shell $
#
# ".gitconfig", ".gitignore" erstellen und konfigurieren
#
# git versionieren
git add .
git commit -a # Editorauswahl: sudo update-alternatives --config
editor
git status
git log --graph --oneline
git lg > git.log

# Github Repository
git pull
git push

# Backup Repository
git remote -v
git push --all backupHD # sichern
git push --all backupUSB
git push --all backupRPI4

# Branch erstellen
git checkout -b feature/a1
git checkout feature/a1
# projekt bearbeiten
git checkout master
git merge feature/a1

git status
git log
git lg
git log --graph --oneline # beenden q
git log --graph --pretty=format:"; %cn; %h; %ad; %s" --
date=relative > git.log
```

### 1.3.6 Markdown Dokumente - Notizen verfassen

Markdown Dokumente / Notizen im Ordner «md/neu.md» erstellen.

```
# Markdown
```

```
<!--ju - Letztes Update: 6-Apr-19 -->
```

```
# Überschrift
```

```
## Überschrift 2
```

```
### Überschrift 3
```

```
## Bild
```

Bilder in pdf speichern, empfehlenswert für \LaTeX.

```
![Logo](images/bildname-001.pdf)
```

```
![Bild](https://cdn.pixabay.com/photo/2019/04/02/04/32/masala-4096891_960_720.jpg)
```

```
## Tabelle
```

```
|**Nr.**|**Begriffe**|**Erklärung**|
|---:|:-----|:-----|
| 1    | a1          | a2          |
| 2    | b1          | b2          |
| 3    | c1          | c2          |
```

## Erste Schritte

scripteBash «projekt.sh», «scripteBash/sed.sh» u. content/metadata.tex anpassen.

```
# Shell $
```

```
cd neue-Notiz
```

```
# scripteBash anpassen
```

```
vi scripteBash/sed.sh
```

```
# codelanguage: HTML5, Python, Bash, C, C++, TeX
```

```
# CMS Server Pfad: https://www.ju1.eu/*
```

```
# Bildformat: pdf -> \LaTeX; svg, png, jpg -> web
```

```
vi projekt.sh
```

```
#-----
```

```
# anpassen
```

```
THEMA="dummy-notizenUbuntu-v03"
```

```
backup_USB="/media/jan/usb/backup/notizenUbuntu"
```

```
#backup_RPI4="smb://rpi4.local/nas/backup/notizenUbuntu"
```

```
backup_HD="/media/jan/virtuell/backup/notizenUbuntu"
```

```
archiv_USB="/media/jan/usb/archiv/notizenUbuntu"
#archiv_RPI4="smb://rpi4.local/nas/archiv/notizenUbuntu"
archiv_HD="/media/jan/virtuell/archiv/notizenUbuntu"

repos_USB="/media/jan/usb/repos/notizenUbuntu"
#repos_RPI4="smb://rpi4.local/nas/repos/notizenUbuntu"
repos_HD="/media/jan/virtuell/repos/notizenUbuntu"
#-----
```

```
vi content/metadata.tex
# Datum, Titel, Autor
```

## Script ausfuehren

```
# Shell $
cd neue-Notiz
./projekt.sh
```

Projekt: Web & \LaTeX Dokumente erstellen unter Ubuntu

- 1) Markdown in (tex, html5) - sed (Suchen/Ersetzen)
- 2) Kopie tex (Pandoc) - tex (Handarbeit)
- 3) Kapitel erstellen, Scripte ausführen
- 4) Beamer
- 5) TEST: Artikel-PDFs erstellen mit latexmk (x\_\*.pdf)
- 6) PDFs erstellen (book-, print-, artikel.pdf) - Archiv (tex)
- 7) Projekt aufräumen
- 8) Git-Version erstellen
- 9) git status und git log --graph --oneline
- 10) git init
- 11) Fotos optimieren (Web, Latex)
- 12) book-pdf-Version erstellen
- 13) Backup (archiv/\*.zip & \*.tar.gz) & (/media/jan/virtuell/backup)
- 14) www
- 15) Beenden?

Eingabe Zahl >\_

### 1.3.7 Bilder optimieren

**JPG Bilder** in den Ordner «img-in/» kopieren.

optimiert Fotos für das Web und die PDF Datei.

### 1.3.8 Backup

```
# Shell $  
cd neue-Notiz  
tar cvzf ../neue-Notiz.tar.gz .
```

## 1.4 Schnellstart

### Befehle

- Git: download <https://git-scm.com/download/win>
- GitHub: Projekt anlegen <https://github.com/ju1-eu/>
- pwd
- git clone «Repository\_name»
- code «file»
- git add «file» oder git add .git remote rm
- git status
- git log oder git lg
- git stash
- git reset

### 1.4.1 Git konfigurieren

```
# Git konfigurieren  
#-----  
git version  
  
# gitconfig anpassen  
git config --global user.name "jan_lap"  
git config --global user.email "esel573@gmail.com"  
  
# gitconfig ansehen  
git config --global --list  
.gitconfig  
.gitattributes  
.gitignore
```

### 1.4.2 Projekt mit GitHub einrichten

Auf GitHub ein Repository erstellen: <https://github.com/ju1-eu/>

- **Repository\_name:** «prj-dummy»
- **Clone with HTTPS:** <https://github.com/ju1-eu/prj-dummy.git>
- weiter mit «Projekt von GitHub downloaden»
- oder
- weiter mit «Projekt lokal einrichten»

### 1.4.3 Projekt von GitHub downloaden

pwd

```
$THEMA="prj-dummy"
rm $THEMA -Recurse -Force
git clone https://github.com/ju1-eu/$THEMA.git
#git clone https://github.com/ju1-eu/$THEMA.git prj1

# weiter mit ...
# Git Workflow / PROJEKT bearbeiten
```

### 1.4.4 Projekt lokal einrichten

pwd

```
# lokales Repository: master
$THEMA="prj-dummy" # Repository_name auf Github erstellen
#rm $THEMA -Recurse -Force
mkdir $THEMA
cd $THEMA
# Repository anlegen
git init # rm -rf .git oder rm .git -Recurse -Force
echo "# prj-dummy" > README.md
git add .
git commit -m"init"
git status
git remote add origin https://github.com/ju1-eu/$THEMA.git
```

```
git push --set-upstream origin master
```

```
# weiter mit ...
```

```
# Git Workflow / PROJEKT bearbeiten
```

## 1.4.5 Git Workflow

### 1.4.5.1 Arbeitsverzeichnis

```
pwd
```

```
# /c/daten/projekte/git/git-kurs/prj
```

```
# C:\daten\projekte\git\git-kurs\prj
```

### 1.4.5.2 PROJEKT bearbeiten

```
git status
```

```
# README.md erstellen
```

```
echo "# prj-dummy" > README.md
```

```
echo "<!-- update: 20-Dez-19 -->" >> README.md
```

```
cat README.md
```

```
# .gitignore erstellen
```

```
echo "# ju update: 20-Dez-19 .gitignore" > .gitignore
```

```
echo "!.gitignore" >> .gitignore
```

```
cat .gitignore
```

### 1.4.5.3 git add - Arbeitsverzeichnis => Stagingbereich

```
git add .
```

```
git add README.md
```

### 1.4.5.4 git commit - Stagingbereich => Repository

```
git commit -m"projekt init"
```

```
git commit
```

```
git commit -a
```

### 1.4.5.5 git push - Repository => Github Repository

```
# Voraussetzung
```

```
$THEMA="prj-dummy" # Repository_name auf Github erstellen
```

```
git remote add origin https://github.com/ju1-eu/$THEMA.git
```

```
git push --set-upstream origin master
```

```
git status
git push
git pull
```

#### 1.4.5.6 git status - sauberes Arbeitsverzeichnis prüfen!

```
git status
  On branch master
  Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

#### 1.4.6 git log - history

```
git status
git log
git log --abbrev-commit --pretty=oneline --graph
git lg
```

#### 1.4.7 Branch neu erstellen - PROJEKT bearbeiten

```
# sauberes Arbeitsverzeichnis prüfen!
git checkout master
git status
# wenn lokale Änderungen vorliegen, wird der Merge abgebrochen
git pull --ff-only

git branch
# Branch neu erstellen
git checkout -b feature/a1
# Feature-Branch zentral sichern
git push --set-upstream origin feature/a1
# Änderungen können zukünftig gesichert werden
git push

# PROJEKT bearbeiten
code README.md
git add .
git commit

# Branch wechseln
git status
git branch
git checkout master
```

```
# Branch zusammenführen
git merge feature/a1

git branch
# nutzlosen Branch löschen
git push -d origin feature/a1
git branch -d feature/a1

# sauberes Arbeitsverzeichnis prüfen!
git status
```

#### 1.4.8 git tag - Version erstellen

```
# sauberes Arbeitsverzeichnis prüfen!
git status

git tag
git tag -a v1.0
# stabile Version

git checkout v1.0
git checkout master
```

#### 1.4.9 git blame - Wer hat was und wann geändert?

```
git blame README.md
```

#### 1.4.10 Datei umbenennen o. löschen - .git löschen

```
git mv file_alt file_neu
git commit -am"Datei umbenannt"
git rm file_neu
git commit -am"Datei gelöscht"
git status

# .git löschen
rm -rf .git
rm .git -Recurse -Force
```

#### 1.4.11 Versionskonflikt lösen

```
# sauberes Arbeitsverzeichnis prüfen!
git status

git branch -a
```



```
# Voraussetzung: Branch muss vorhanden sein
# siehe Branch neu erstellen
git checkout feature/a1

# PROJEKT bearbeiten auf feature/a1 Branch
code README.md
# file: README.md bearbeiten
git commit -am"README.md bearbeitet"
git push
git diff master feature/a1

git checkout master

# PROJEKT bearbeiten auf master Branch
code README.md
# file: README.md bearbeiten
git commit -am"README.md bearbeitet master"
git push
git status

# zusammenführen mit master-Branch
git merge feature/a1
# Fehler, weil auf unterschiedlichen Branches gearbeitet wurde
#-----
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
#-----
# Konflikt bearbeiten
code README.md
# file bereinigen: <<<<<<, ===== und >>>>>>

git commit -am"Konflikt behoben"
git push

git branch -a
git push -d origin feature/a1
git branch -d feature/a1

# sauberes Arbeitsverzeichnis prüfen!
git status
```

#### 1.4.12 Lokales Wiederherstellen

```
# Änderungen zwischenspeichern
git stash --include-untracked
# Letzte gespeicherte Änderungen zurückholen
git stash pop
```

pwd

```
# PROJEKT bearbeiten
echo "neu" > file.txt
git add .
git commit -m"neu"
echo "Fehler" >> file.txt
# Achtung: noch kein git commit!
git status
```

```
# Möglichkeit 1
git log
cat .\file.txt
git checkout HEAD file.txt
cat .\file.txt
git status
```

```
# Möglichkeit 2
git log
cat .\file.txt
git reset --hard HEAD
cat .\file.txt
git status
```

### 1.4.13 Repository Wiederherstellen

pwd

```
# PROJEKT bearbeiten
echo "Fehler 2" >> file.txt
git commit -am"Fehler provoziert"
# Achtung: noch kein git push!
git status
```

```
# Möglichkeit 1
git log
git reset --hard d446456
git log
git status
```

```
# Möglichkeit 2
git log
git revert c3b6fab
git log
git status

git push
git status
```

### 1.4.14 Git Backup - Repository ohne einen Workspace

Backup auf USB-Stick

pwd

```
# backup Repository: backup_wlap/master
# anpassen
$PFAD="E:/backup-Repos/notizenWin10"
$THEMA="prj-dummy"
$REPO="backup_wlap"

#cd $THEMA
git clone --no-hardlinks --bare . $PFAD/$THEMA.git
git remote add $REPO $PFAD/$THEMA.git
# gelegentlich sichern
git status
git push --all $REPO

# löschen
rm -rf $PFAD/$THEMA
rm $PFAD/$THEMA -Recurse -Force
```

Backup auf Raspberry Pi

pwd

```
# backup Repository: backup_rpi4/master
# anpassen
$PFAD="//RPI4\usbstick\backup-Repos\notizenWin10"
$THEMA="prj-dummy"
$REPO="backup_rpi4"

#cd $THEMA
git clone --no-hardlinks --bare . $PFAD/$THEMA.git
git remote add $REPO $PFAD/$THEMA.git
# gelegentlich sichern
```

```
git status
git push --all $REPO

# löschen
rm -rf $PFAD/$THEMA
rm $PFAD/$THEMA -Recurse -Force
```

### 1.4.15 lokales Repository clonen

```
pwd
cd C:\daten\projekte\git\git-kurs\prj

# löschen
rm -rf $THEMA
rm $THEMA -Recurse -Force

$PFAD="E:/backup-Repos/notizenWin10"
$THEMA="prj-dummy"
$REPO="backup_wlap"

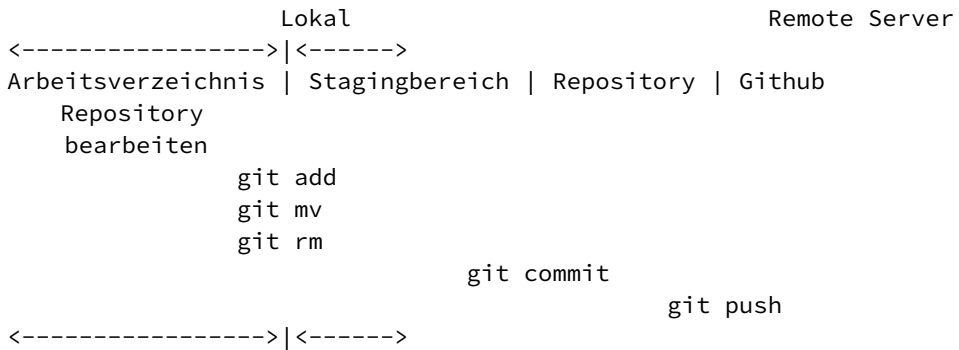
git clone $PFAD/$THEMA.git
cd $THEMA

# oder

$PFAD="\\RPI4\usbstick\backup-Repos\notizenWin10"
$THEMA="prj-dummy"
$REPO="backup_rpi4"
git clone $PFAD/$THEMA.git
cd $THEMA
```

## 1.5 Git Workflow - Bereiche

- Datei bearbeiten
- Datei hinzufügen
- Datei Umbenennen
- Datei löschen



## 1.6 Wiederherstellen

### 1.6.1 Ordner für Experimente erstellen - löschen

```
# Shell $
cd projekt
mkdir -p repoWork repoNeu repoAlt
# löschen
rm -rf repoWork repoNeu repoAlt
```

### 1.6.2 bestehendes Repository clonen

```
cd neue-Notiz # ? Repository
git clone . ../repoWork
```

### 1.6.3 Arbeitsverzeichnis bearbeiten

#### File bearbeiten 1

```
# Shell $
cd repoWork
vi test.md
# file
Basis

# git versionieren
git add .
git commit -a
git status
```

#### File bearbeiten 2

```
# Shell $
vi test.md
# file
Basis
2) Version

# git versionieren
git commit -a
git status
```

### File bearbeiten 3

```
# Shell $
vi test.md
# file
Basis
2) Version
3) Version

# git versionieren
git commit -a
git status
git log --graph --oneline
```

## 1.6.4 Wiederherstellen: Repository in ein temp. Verzeichnis klonen

```
# Shell $
cd repoWork
git clone . ../repoNeu
git clone . ../repoAlt
```

## 1.6.5 Wechsel auf den gewünschten Git-Branch

```
# Shell $
cd ../repoNeu/
git stash
git log --graph --oneline
* 85f27a3 (HEAD -> master) version 3
* 22355e7 Vesion 2
* 23e67c4 Version 1
* 5c1d6a7 (origin/master, origin/HEAD) aufgeräumt
* 138684a prj
* 2af6ca2 Projekt init
# version2
git reset --hard 22355e7
```

## 1.6.6 Verschiebe .git in den Workspace der alten Versionsverwaltung

```
# Shell $  
git archive master | tar -x -C ../repoAlt/
```

## 1.6.7 Ergebnis prüfen

```
# Shell $  
cd projekt  
kdiff3 repoAlt/ repoNeu/Repository
```

## 1.7 Repository Versionen vergleichen

### 1.7.1 Ordner für Experimente erstellen - löschen

```
# Shell $  
cd projekt  
mkdir -p vers_Lokal vers_Github vers_backupHD vers_backupUSB  
vers_backupRPI4  
# löschen  
rm -rf vers_Lokal vers_Github vers_backupHD vers_backupUSB  
vers_backupRPI4
```

### 1.7.2 lokales Repository

```
# Shell $  
# repository clonen  
cd repoWork  
VERZ="vers_Lokal"  
git clone . ../$VERZ  
  
# build - Versionen erstellen  
VERZ="vers_Lokal"  
#tar cvzf ../$VERZ.tgz .  
ID=$(git rev-parse --short HEAD) # git commit (hashwert)  
timestamp=$(date +"%d%m%y") # Datum: 260619  
# 260619_Verz_v0b61478.tgz  
archiv=$timestamp'_ '$VERZ'_v'$ID  
tar cvzf ../$archiv.tgz .  
cd ..
```

### 1.7.3 Github Repository

origin/master

```
# Shell $
# Github repository clonen
REPOSITORY="dummy-notizenUbuntu-v03"
ADRESSE="https://github.com/ju1-eu"
cd $REPOSITORY
git clone $ADRESSE/$REPOSITORY.git .
# oder
REPOSITORY="dummy-notizenUbuntu-v03"
ADRESSE="git@github.com:ju1-eu"
cd $REPOSITORY
git clone $ADRESSE/$REPOSITORY.git .

# build - Versionen erstellen
VERZ="vers_Github"
#tar cvzf ../$VERZ.tgz .
ID=$(git rev-parse --short HEAD) # git commit (hashwert)
timestamp=$(date +"%d%m%y") # Datum: 260619
# 260619_Verz_v0b61478.tgz
archiv=$timestamp'_ '$VERZ'_v'$ID
tar cvzf ../$archiv.tgz .
cd ..
```

## 1.7.4 Backup Repository

```
# Shell $
# Backup Repository: backupHD/master
HD="/media/jan/virtuell/repos/notizenUbuntu"
REPOSITORY="dummy-notizenUbuntu-v03"
LESEZEICHEN="backupHD"
git clone --no-hardlinks --bare . $HD/$REPOSITORY.git
git remote add $LESEZEICHEN $HD/$REPOSITORY.git
git push --all $LESEZEICHEN

# build - Versionen erstellen
VERZ="vers_backupHD"
#tar cvzf ../$VERZ.tgz .
ID=$(git rev-parse --short HEAD) # git commit (hashwert)
timestamp=$(date +"%d%m%y") # Datum: 260619
# 260619_Verz_v0b61478.tgz
archiv=$timestamp'_ '$VERZ'_v'$ID
tar cvzf ../$archiv.tgz .
cd ..
```



```
# backup Repository: backupUSB/master
# anpassen
USB="E:/repos/notizenUbuntu"
REPOSITORY="dummy-notizenUbuntu-v03"
LESEZEICHEN="backupUSB"
git clone --no-hardlinks --bare . $USB/$REPOSITORY.git
git remote add $LESEZEICHEN $USB/$REPOSITORY.git
git push --all $LESEZEICHEN

# build - Versionen erstellen
VERZ="vers_backupUSB"
#tar cvzf ../$VERZ.tgz .
ID=$(git rev-parse --short HEAD) # git commit (hashwert)
timestamp=$(date +"%d%m%y") # Datum: 260619
# 260619_Verz_v0b61478.tgz
archiv=$timestamp'_'$VERZ'_v'$ID
tar cvzf ../$archiv.tgz .
cd ..

# backup Repository: backupRPI4/master
# anpassen
RPI4="//RPI4\nas\repos\notizenUbuntu"
REPOSITORY="dummy-notizenUbuntu-v03"
LESEZEICHEN="backupRPI4"
git clone --no-hardlinks --bare . $RPI4/$REPOSITORY.git
git remote add $LESEZEICHEN $RPI4/$REPOSITORY.git
git push --all $LESEZEICHEN

# build - Versionen erstellen
VERZ="vers_backupRPI4"
#tar cvzf ../$VERZ.tgz .
ID=$(git rev-parse --short HEAD) # git commit (hashwert)
timestamp=$(date +"%d%m%y") # Datum: 260619
# 260619_Verz_v0b61478.tgz
archiv=$timestamp'_'$VERZ'_v'$ID
tar cvzf ../$archiv.tgz .
cd ..
```

## 1.7.5 Ergebnis prüfen

```
# Shell $
cd projekt
# Verzeichnisse vergleichen
# vers_Lokal vers_Github vers_backupHD vers_backupUSB
  vers_backupRPI4
```

```
kdiff3 vers_Lokal/ vers_Github/  
kdiff3 vers_Lokal/ vers_backupHD  
kdiff3 vers_backupHD vers_backupUSB vers_backupRPI4  
# files vergleichen  
kdiff3 vers_Lokal/Readme.md vers_Github/Readme.md
```

## 1.7.6 Projekt Inhalt

```
# Shell $  
cd projekt  
ls -lh *gz  
# Inhalt  
260619_vers_Github_v5c1d6a7.tgz  
260619_vers_Lokal_v5c1d6a7.tgz
```

## 1.7.7 build - Versionen erstellen

```
# Shell $  
file="MD5-Hash.txt"  
printf "# -----\\n" > $file  
printf "# build - Versionen\\n" >>  
$file  
printf "# MD5-Hash: Datum_Verzeichnis_vGit-Hashwert.tgz\\n" >>  
$file  
printf "# -----\\n\\n" >> $file  
  
# hashwert erstellen  
md5sum 260619_vers_Github_v5c1d6a7.tgz >> $file  
md5sum 260619_vers_Lokal_v5c1d6a7.tgz >> $file  
  
# build - Versionen  
vi MD5-Hash.txt
```

## 1.8 Markdown - Spickzettel

### 1.8.1 Quellenangabe

Zitat: vgl. [Mon16] u. [Kof17]

### 1.8.2 Listen

#### ungeordnete Liste

- a
- b
  - bb
- c

#### Sortierte Liste

1. eins
2. zwei
3. drei

#### Sortierte Liste

- a) a
- b) b
- c) c

### 1.8.3 Anführungszeichen

«Anführungszeichen»

## 1.8.4 Bilder - Abbildungen



## 1.8.5 Tabelle

Nr.	Begriffe	Erklärung
1	a1	a2
2	b1	b2
3	c1	c2
4	a1	a2

## 1.8.6 Mathe

$$[V] = [\Omega] \cdot [A] \text{ o. } U = R \cdot I \text{ o. } R = \frac{U}{I}$$

**Matheumgebung:**

$$\sum_{i=1}^5 a_i = a_1 + a_2 + a_3 + a_4 + a_5$$

## 1.8.7 Texthervorhebung

**Fett** oder *Kursiv*

## 1.8.8 Code

```
#include <stdio.h>
int main(void) {
    printf("Hallo Welt!\n");
    return 0;
```

}

### 1.8.9 Links

<https://google.de> oder Google

### 1.8.10 Absätze

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: «Dies ist ein Blindtext» oder «Huardest gefburn»? Kjift - mitnichten! Ein Blindtext bietet mir wichtige Informationen.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: «Dies ist ein Blindtext» oder «Huardest gefburn»? Kjift - mitnichten! Ein Blindtext bietet mir wichtige Informationen.

# Literaturverzeichnis

- [Kof17] M. Kofler. *Linux: Das umfassende Handbuch von Michael Kofler. Für alle aktuellen Distributionen (Desktop und Server)*. Rheinwerk Computing. Rheinwerk Verlag GmbH, 2017. ISBN: 9783836258548.
- [Mon16] Simon Monk. *Das Action-Buch für Maker: Bewegung, Licht und Sound mit Arduino und Raspberry Pi - Experimente und Projekte*. 1. Auflage. Heidelberg: dpunkt, 29. Sep. 2016. 360 Seiten. ISBN: 978-3-86490-385-4.